From: AAAI Technical Report WS-97-08. Compilation copyright © 1997, AAAI (www.aaai.org). All rights reserved.

Heuristic Real-Time Dynamic Programming (Preliminary Results)

Blai Bonet and Héctor Geffner

Departamento de Computación
Universidad Simón Bolívar
Aptdo. 89000, Caracas 1080-A, Venezuela
{bonet,hector}@usb.ve

Abstract

Building on the work of Korf on real-time search, and of Barto, Bradtke and Singh on real-time dynamic programming, we discuss three features that we have found crucial for the solution of large MDP's; namely, selective updates, selective lookahead, and good heuristic functions. We also present an RTDP algorithm that exhibits these features and assess its performance over three types of problems.

Introduction

We have become interested in real-time algorithms for solving large Planning and Markov Decision Problems in the last year, where we have been working on schemes for combining planning and execution. Using a real time search algorithm (a variation of Korf's LRTA* (Korf 1990)) together with a domain independent heuristic function designed for 'classical' planning tasks, we were able to develop an algorithm called ASP[1] that interleaves planning and execution, and is fast, robust and very powerful (e.g., it solves the hard planning problems in (Kautz & Selman 1996) in a few seconds; see (Bonet, Loerincs, & Geffner 1997)). Although ASP does not produce optimal plans, the plans obtained in the domains we have considered are not far from optimal and can be improved after repeated trials.

We are currently working on extensions of this approach for dealing with more general planning problems, namely, those that can be expressed as *shortest path MDP's;* i.e., Markov Decision Problems in which a set of absorbing goal-states is to be reached through an action 'path' of minimum expected cost (Barto, Bradtke, & Singh 1995). The reason we find this extension feasible and promising is that:

- Planning problems are shortest-path MDP's
- A 'planning algorithm' like ASP solves problems that are several orders of magnitude larger than problems handled by current MDP algorithms
- ASP (like Korf's LRTA*) is a special case of Barto's *et al* **Real Time Dynamic Programming** (RTDP) algorithms that are suitable for solving general MDP's (Barto, Bradtke, & Singh 1995)

We have been exploring the use of a more general ASP like search algorithm for shortest-path MDP problems, taking into account both the costs and probabilities associated with the actions. The new algorithm is an RTDP algorithm that differs from ASP at just one point. Rather than computing the value $Q(x,a)$ of an action a in a state x as $1+V(x')$, where $V(x')$ is the value of the state that results from action a in x, $Q(x,a)$ is determined from the stochastic DP equation (Barto, Bradtke, & Singh 1995):

$$Q(x,a) = c(x,a) + \sum_{x'} p_a(x'|x)V(x') \qquad (1)$$

that reflects both costs $c(x,a)$ and probabilities $p_a(x'|x)$. Except for this single change, the RTDP algorithm below is identical to ASP.

The RTDP Algorithm

The RTDP algorithm is a real time search algorithm (Korf 1990) in the style of the algorithms used in two-player games like chess that perform local searches before moving. A single trial of the algorithm interleaves real and simulated moves until a goal state is reached as follows:

The RTDP algorithm

1. *Lookahead Search:* From the current state x, perform n *simulated* moves.
2. *More Local Search:* Repeat lookahead search m times (always starting from the current state x).
3. *Move:* Perform a *real* move

The simulated moves provide the local search that precedes the real moves. Both types of moves involve the following steps:

1. *Expand:* Calculate $Q(x,a)$ for each action a in x using equation (1). Initially, the values $V(x)$ are set to $h(x)$, where h is a suitable heuristic function.
2. *Update:* Update the estimated state values as:

$$V(x) \leftarrow \min_a Q(x,a)$$

[1] The code of ASP and some sample problems are available at http://www.eniac.com/~bbonet.

3. *Go:* Apply the action a that has a minimum $Q(x,a)$ value, breaking ties randomly.

The only difference between real and simulated moves is that the former affect the current state of the system while the later only affect the current state of the system in the simulation.

The local search above differs from the local search in Korf's LRTA* and related algorithms (e.g., (Dearden & Boutilier 1994)) in two ways. First, the search within the local space is guided by the value function V and is not exhaustive (it visits at most $m \times n$ states). Second, during the lookahead, the value function of *every state visited* is updated (that's the reason it makes sense to repeat the lookahead search m times). These two features have a considerable effect on performance and both are considered in (Barto, Bradtke, & Singh 1995) (see below). Also, as proven in (Barto, Bradtke, & Singh 1995), after a sufficient number of trials (see also (Korf 1990)), this type of algorithm is guaranteed to converge to the optimal 'path' when the heuristic function $h(x)$ used to initialize the values of the states is optimistic. In the experiments below, however, only single trial versions of the algorithm are considered.

The Power of Heuristic RTDP Algorithms

Planning problems are simple MDP's as they ignore probabilities and costs yet that alone does not explain the performance of algorithms like ASP that appear to handle very large problems (e.g., 10^{25} states) producing solutions whose costs do not exceed the optimal costs in more than 50%. We believe that the explanation for these results rests mainly in three features; namely, **real time updates, good heuristic functions** and **selective lookahed**:

- **real time DP algorithms** (and other asynchronous DP algorithms) do not consider the whole space of the problem but only the states that result from the actions of the agent. When the initial heuristic estimates are *optimistic,* this is guaranteed to focus the updates on states that are relevant (Korf 1990; Barto, Bradtke, & Singh 1995). A different method for isolating the part of the space that is relevant is given in (Dean *et al.* 1993).
- the selectivity of RTDP algorithms is further increased by the use of **good heuristic functions** for initializing the values of the states when they first are visited (Korf 1990). These heuristics can be crafted by hand or can be derived from suitable representations (Dearden & Boutilier 1994). In the planning problems we considered, the heuristic function was derived automatically from a Strip-like representation and made a huge difference in the results in spite of not being even optimistic (Bonet, Loerincs, & Geffner 1997).
- the imperfections in the heuristic function can often be smoothed by **looking ahead** far enough. Yet, local search methods are often exhaustive (with the exception of some pruning; see (Korf 1990; Dearden & Boutilier 1994)) and thus grow exponentially in the depth of the local space. An alternative that paid off in our experience was to search the local space by hill climbing making use of the estimated state values to guide the search. This actually corresponds to the combination of real and simulated moves discussed in (Barto, Bradtke, & Singh 1995) and in other places (e.g., (Sutton 1990)). Also, by updating the state values as the local space is explored and by repeating this process a number of times, useful information about the local search space can be gathered that may lie at considerable depth in the search graph (see below).

problem	opt.	steps	time	visited	space
blocks_1	6	11	0.40	195	$4.6 \cdot 10^6$
blocks_2	9	14	0.65	306	$8.2 \cdot 10^8$
blocks_3	14	23	3.17	645	$6.6 \cdot 10^{13}$
blocks_4	18	31	12.97	1034	$1.4 \cdot 10^{19}$

Table 1: RTDP with minimal lookahead ($m = n = 1$) on four block world planning problems

Some Experimental Results

The experiments below illustrate the impact on performance of the three main features discussed: **real time updates, good heuristic functions** and **selective lookahed.**

The first set of experiments illustrate the synergy that results from the use of a good heuristic function with an RTDP algorithm. Table 1 shows the results of the RTDP algorithm with the planning heuristic reported in (Bonet, Loerincs, & Geffner 1997) and a *minimal lookahead* (i.e., $n = m = 1$) over four block world planning problems involving 9, 11, 15 and 19 blocks respectively (see (Kautz & Selman 1996)). In all these cases the problems are solved very fast and only a tiny fraction of the state space is visited (columns 5 and 6). The cost of the solutions (column 3) in almost all these cases is 50% above the optimal costs (column 2). These solutions can be improved in two ways: by either increasing the lookahead parameters n and m, or performing repeated trials (see (Bonet, Loerincs, & Geffner 1997) for the results in both cases). In any case, it's clear that for these type of problems, optimal but exhaustive techniques like value iteration are not applicable. Moreover, the same RTDP algorithm with a non-informative heuristic function, $h(x) = 0$, does not find solutions to *any* of the problems.

The second experiment illustrates the value of selective lookahead (Table 2) as opposed to exhaustive lookahead (Table 3) over an instance of the 8-puzzle. In the first scheme, the best tradeoffs are obtained for approximated 18 simulated moves in each local search for a fixed number m of local searches equal to 10. In this case the solution is found after 3.25 seconds and has a cost of 31 (optimal cost is 24). Table 3 shows the result of replacing the selective local search by an

	steps	time
$n = 1$	397.05	3.32
$n = 2$	124.95	2.05
$n = 3$	146.95	3.20
$n = 4$	103.65	3.07
$n = 5$	99.25	3.45
$n = 6$	79.65	3.27
$n = 7$	66.25	3.32
$n = 8$	55.25	3.07
$n = 9$	49.60	2.97
$n = 10$	50.00	3.45
$n = 12$	33.55	2.52
$n = 14$	38.30	3.30
$n = 16$	36.90	3.62
$n = 18$	31.80	3.25
$n = 20$	34.15	3.87
$n = 22$	34.00	4.17
$n = 24$	35.05	4.85

Table 2: RTDP with selective lookahead ($m = 10$ and variable n) over an instance of the 8-puzzle

	steps	time
$d = 1$	349.7	0.55
$d = 2$	188.0	0.80
$d = 3$	135.0	1.75
$d = 4$	110.0	3.90
$d = 5$	114.5	11.00
$d = 6$	81.2	22.38
$d = 7$	69.2	53.13
$d = 8$	65.2	141.60
$d = 9$	—	—
$d = 10$	—	—

Table 3: RTDP with exhaustive lookahead and depth d over same instance of 8-puzzle

exhaustive local search up to a depth represented by the parameter d. Clearly the time taken to explore this local space grows exponentially, and even with a depth $d = 8$ and time equal to 141 seconds, we don't get a solution of the same quality.[2]

Finally, Table 4 shows RTDP solutions to a stochastic 'race track' problem as the one reported in (Barto, Bradtke, & Singh 1995), where the goal is to go around the track in a minimal number of actions (x and y accelerations). The problem we considered involves around 100,000 states that result from the different positions and velocities of the car, and can be solved by *value iteration* in 12 minutes. The expected cost of the optimal solution is 20. Columns 2-3 and 4-5 show the number of steps and seconds taken by RTDP solutions involving a non-informative heuristics $h = 0$ and an informative heuristics (a composition of aerial distances), for different numbers n of simulated moves

[2]The number of steps reported in each table are averages as different runs of the algorithms yield different results due to the random tie breaking mechanism.

	$h = 0$		$h \neq 0$	
	steps	time	steps	time
$n = 1$	—	—	14.42	535.2
$n = 5$	148.7	8.59	2.24	27.7
$n = 10$	104.7	10.12	3.35	32.6
$n = 20$	103.1	17.80	4.34	26.4
$n = 40$	69.7	23.31	7.74	30.6
$n = 60$	64.0	30.12	9.79	31.2

Table 4: Solutions to a stochastic race track problem with different heuristics

in each local search (the number m of local searches is fixed to 30). Again there is a number of simulated moves ($n = 20$ when using an informative heuristics) that provides an optimal tradeoffs between solution quality and time, and the quality of the heuristic function for the initialization of the state values makes a significant difference.

Summary

Real time dynamic programming algorithms appear to scale to large MDP's when suitable local search strategies are used, all and only visited states are updated, and good heuristics are available. The first two element are built in the RTDP algorithm we have used (in the form of a hill-climbing local search that updates state values upon real or simulated moves), while the third element has to be crafted for each application (although as shown in (Dearden & Boutilier 1994; Bonet, Loerincs, & Geffner 1997) suitable action representations may allow the automatic computation of heuristics in many cases of interest).

References

Barto, A.; Bradtke, S.; and Singh, S. 1995. Learning to act using real-time dynamic programming. *Artificial Intelligence* 72:81–138.

Bonet, B.; Loerincs, G.; and Geffner, H. 1997. A robust and fast action selection mechanism for planning. In *Proceedings of AAAI-97*. MIT Press.

Dean, T.; Kaelbling, L.; Kirman, J.; and Nicholson, A. 1993. Planning with deadlines in stochastic domains. In *Proceedings AAAI-93*, 574–579. MIT Press.

Dearden, R., and Boutilier, C. 1994. Integrating planning and execution in stochastic domains. In *Proceedings of UAI-94*, 162–169. Morgan Kaufmann.

Kautz, H., and Selman, B. 1996. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of AAAI-96*, 1194–1201. Protland, Oregon: MIT Press.

Korf, R. 1990. Real-time heuristic search. *Artificial Intelligence* 42:189–211.

Sutton, R. 1990. Integrated architectures for learning, planning and reacting based on approximating dynamic programming. In *Proceedings of ML-90*, 216–224. Morgan Kaufmann.